# An Agent-Based Algorithm for Detecting Community Structure in Networks

Maxwell Young [*] Jennifer Sager [†] Gábor Csárdi [‡] Péter Hága [§]

**Abstract**

We present a simple stochastic agent-based community finding algorithm. Our algorithm is tested on network data from the Zachary karate club study, data from Victor Hugo's *Les Miserables* [1], and data obtained from a musical piece by J.S. Bach. In all three cases, the algorithm partitions the vertices of the graph sensibly.

## 1 Introduction

Assume that we have a data set with a graph representation $G = (V, E)$ where the vertices represent objects and edges represent some relationship between these objects [2]. The goal of a community-finding algorithm is to partition the vertices of the graph into groups which, hopefully, provide insight into the structure of the relationships in $G$.

Determining community-structure is arguably not a well-defined problem. It is not obvious how one should rigorously define the notion of a "community". Consequently, interpreting the results of a community-finding algorithm can be a qualitative process in which one can only hope to uncover a meaningful partitioning of the vertices. However, there is an abundance of research showing that many community-finding algorithms provide similar and sensible results. Moreover, these algorithms can be tested on real world data and, therefore, some measure of correctness can be obtained. Together, these points lend credibility to the idea that such algorithms are indeed consistently measuring some structural property of the graph in question.

Many community-finding algorithms have been proposed; see work done by Newman [2,3,4,6]. In particular, the algorithms in [2,3,4] have performed well both in terms of the communities found and the running time complexity. The work presented in this paper differs from these algorithms in two ways. First, ours is an agent-based approach where, as will be explained later, collective exploration and analysis of the given graph

[*] Department of Computer Science, University of New Mexico, Albuquerque, NM 87131-1386; email: youn@cs.unm.edu. Corresponding author.

[†] Department of Computer Science, University of New Mexico, Albuquerque, NM 87131-1386; email: sagerj@cs.unm.edu. Corresponding author.

[‡] Department of Biophysics, KFKI Research Institute for Particle and Nuclear Physics of the Hungarian Academy of Sciences, Budapest, Hungary; email: csardi@rmki.kfki.hu.

[§] Department of Physics of Complex Systems, Eötvös University, Budapest, Hungary; email: haga@complex.elte.hu.

[1] Data for the Zachary karate club study and *Les Miserables* kindly provided by Mark Newman.

[2] $G$ may be directed and possess edge-weights. Our algorithm was designed and tested only on undirected, unweighted graphs. While our algorithm should be able to handle directed graphs, we have no results by which to gauge its performance on such problem instances

yields the final results. Second, our algorithm is stochastic and so the algorithm can exhibit variability over the same input; whether this is a useful attribute is open to discussion.

## 2 The Algorithm

In this section, we provide a general description of our algorithm as well as give some motivation for certain aspects of the algorithm. Later on, we give a more detailed account of the algorithm.

### 2.1 A General Description

Our algorithm employs biased random walks on a graph by many agents which we will refer to as "ants" [3]. Ants are initially placed on vertices chosen uniformly at random from $V$. These ants then perform a random walk of prespecified length and keep a list of the vertices they traverse; this is the "exploration phase" of the algorithm. This random walk is biased in the sense that an ant may not traverse an edge they most recently crossed unless there is no alternative; in this way, we deter backtracking. Once each ant has completed its walk, they convene and implement a voting process, detailed in the next section, by which $V$ is partitioned. After this voting process has occurred, communities have been formed and the results may be output by the algorithm. However, if we know how many communities are desired, then the algorithm can enter into a "cleanup" stage whereby these communities are merged into larger communities until the appropriate number of communities has been obtained.

As previously mentioned, a "community" is not well-defined; however, this does not prevent us from modeling our algorithm after a qualitative understanding of the term. Intuitively, we would expect there to be a higher degree of connectedness between those nodes within a community compared to those nodes that are not. Those regions of low connectivity between nodes could be viewed as "bridge areas" linking communities to one another. Consequently, the motivation for deterring backtracking is that it forces an ant to cross a so-called bridge area and enter into a community. Once inside a community, an ant will be confronted with many more edges that it may choose to follow. Due to this high-connectivity, it is reasonable to argue that the ant is likely to spend a significant portion of its walk-length inside the community because there are more ways to stay within the community than there are to leave it.

### 2.2 The Algorithm in Detail

There are four parameters, one of which is optional, that are set prior to running the algorithm on a problem instance:

- *Population Size* $\rho$: the number of ants that will explore $G$.
- *Walk-Length* $l$: the number of edges an ant traverses before it ends its biased random walk.

[3] However, our algorithm does not fall into the traditional ant-colony optimization paradigm. First, we have no objective function by which to measure a candidate solution. Second, the traditional use of "pheromone trails" is applied differently under this algorithm.

- *Voting Cut-Off $c$*: the value used to discern whether or not a two vertices belong to the same community.
- *Number of Desired Communities $\lambda$*: this is an optional parameter that allows the algorithm to output a desired number of communities during the clean-up phase.

We now give a detailed description of how our algorithm works:

**Algorithm**

1. Each ant is placed on a vertex chosen uniformly at random.
2. Each ant performs a biased random walk and maintains a list of those vertices it traverses.
3. Once each ant has finished its walk, a voting phase begins in order to decide how to partition $V$. For all pairs of vertices $u$ and $v$, the algorithm calculates the number of ants that traversed both $u$ and $v$, call this number $A$. The algorithm also calculates the number of ants that traversed one of $u$ or $v$ (or both), call this value $B$. $u$ and $v$ are merged into the same component if $\frac{A}{B} \geq c$.

   Of course, it is possible that $u$ has already been merged with some other set $S_u$ of vertices prior to its comparison to $v$ (the same is true of $v$). If $u$ and $v$ are merged, then the new community is actually $S_u \cup S_v$; that is, both $u$ and $v$ *and* their respective sets are combined [4].
4. *Optional Clean-up Phase:* communities are sorted by size from smallest to largest. A smaller community $C_i$ is merged into another community of equal or greater size based on the number of *edges it shares with such a community.* We say $C_i$ shares $k$ edges with community $C_j$ if there are $k$ edges linking vertices in $C_i$ to vertices in $C_j$. For instance, if $C_i$ shares 10 edges with community $C_g$, 8 edges with community $C_h$, and 0 edges with all other communities, then $C_i$ and $C_g$ are combined into a new larger community. This procedure is executed until $\lambda$ communities are formed.
5. The partitioning of $V$ is returned.

## 3 Our Results

Our algorithm was tested on three data sets. The first is the real-world data from the Zachary karate club study examined in [2,3,4]. This is a good test case for our algorithm because we already know how the actual partitioning occurred and how other algorithms have performed on this data. Figure 1 demonstrates the actual division undergone by the club.

We first ran our algorithm without the clean-up phase and Figure 2 depicts the results. Demonstrating the resulting communities pictorially is difficult to do without color[5]; therefore, we have also provided the data in Table 1 which gives the partitioning by vertex number along with some other useful information. Note that we get 11 partitions

[4] We employed the use of disjoint-set data structures with union-find operations with path compression in order to efficiently facilitate this stage of the algorithm.

[5] Please view this paper via Adobe Reader in order to see the details of the shading

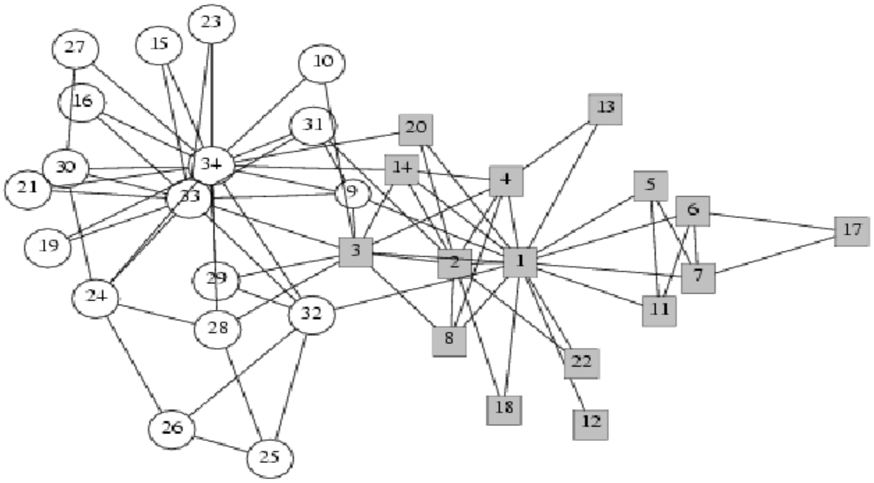

Figure 1: Actual Partitioning of the Karate Club. This image was taken from Mark Newman's website http://www-personal.umich.edu∼mejn/networks/

instead of the desired 2. While this result is not ideal, it is certainly encouraging. First of all, while we have seemingly not merged enough, the partitioning our algorithm provides has done no incorrect merging so far. That is, while more merging is needed to arrive at two groups, no mergings done so far need to be undone in order to obtain the real-world result. Second, it appears that the algorithm is grouping those nodes that share a higher degree of connectedness. For example, the algorithm consistently returned nodes 5, 6, 7, 11, and 17 as a community [6]. This data was obtained with $\rho = 200$, $l = 11$, and $c = 0.75$.

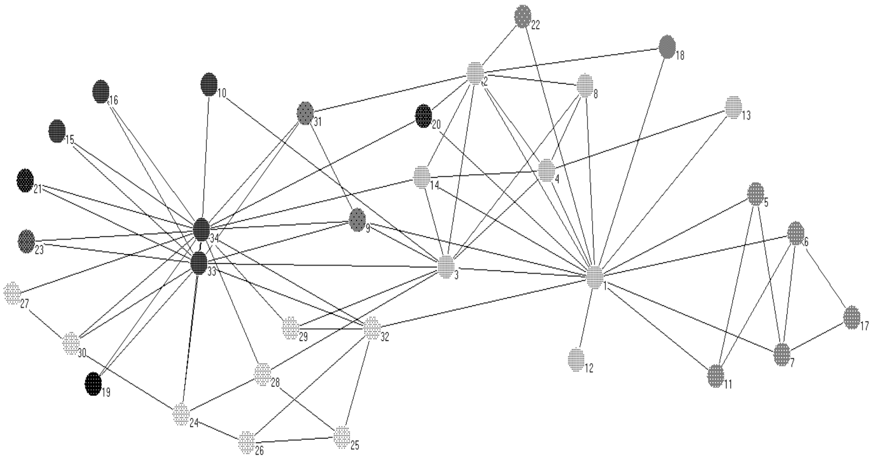

Figure 2: Our Algorithm without Cleaning on the Zachary Karate Club Data

[6] Obviously, more testing needs to be done - testing that we could not perform with the time given to us. However, the algorithm did return these groups fairly consistently.

| Karate Communities | | |
|---|---|---|
| Cleaned Community | Uncleaned Community | Members |
| 1 | 1 | 1, 2, 3, 4, 8, 12, 13, 14 |
| | 2 | 5, 6, 7, 11, 17 |
| | 3 | 22 |
| | 4 | 18 |
| | 5 | 20 |
| 2 | 6 | 9, 31 |
| | 7 | 10, 15, 16, 33, 34 |
| | 8 | 23 |
| | 9 | 21 |
| | 10 | 19 |
| | 11 | 24, 25, 26, 27, 28, 29, 30, 32 |

Table 1 - The eleven communities found by our algorithm using the karate club data.

We ran our algorithm with the same parameter values and the cleaning option with $\lambda = 2$. The resulting partitioning is correct in the sense that the split achieved by our algorithm is exactly that which occurred during the Zachary study; identical to the split demonstrated in Figure 1. Moreover, our algorithm returned fairly consistent results when run numerous times on this data set. For example, out of ten trials, it returned the correct 16-18 split three times, a 17-17 split 3 times, a 19-15 split 2 times, and an 11-23 split once. With the exception of the 11-23 split, the nodes in contention were consistently 3, 9, and 31.

The next data set that we tested our algorithm on was the *Les Miserables*[7]; our partitioning is given in Figure 3. This is not real-world data and, consequently, there is no "correct" partitioning. However, we are able to compare our results to those achieved by Newman; this gives us some idea of how well our algorithm is performing. Due to the difficulty in depicting our partitioning pictorially, we have also provided our results in a table in the appendix at the end of this report. This table displays our communities alongside those found by one of Newman's algorithms. The results of running our algorithm on the *Les Miserables* data are surprisingly close those achieved by Newman. Note that in many of the communities, the majority of the members are identical.

[7] Characters are represented by labeled vertices. Vertices share an edge if the associated characters appeared in the same scene.

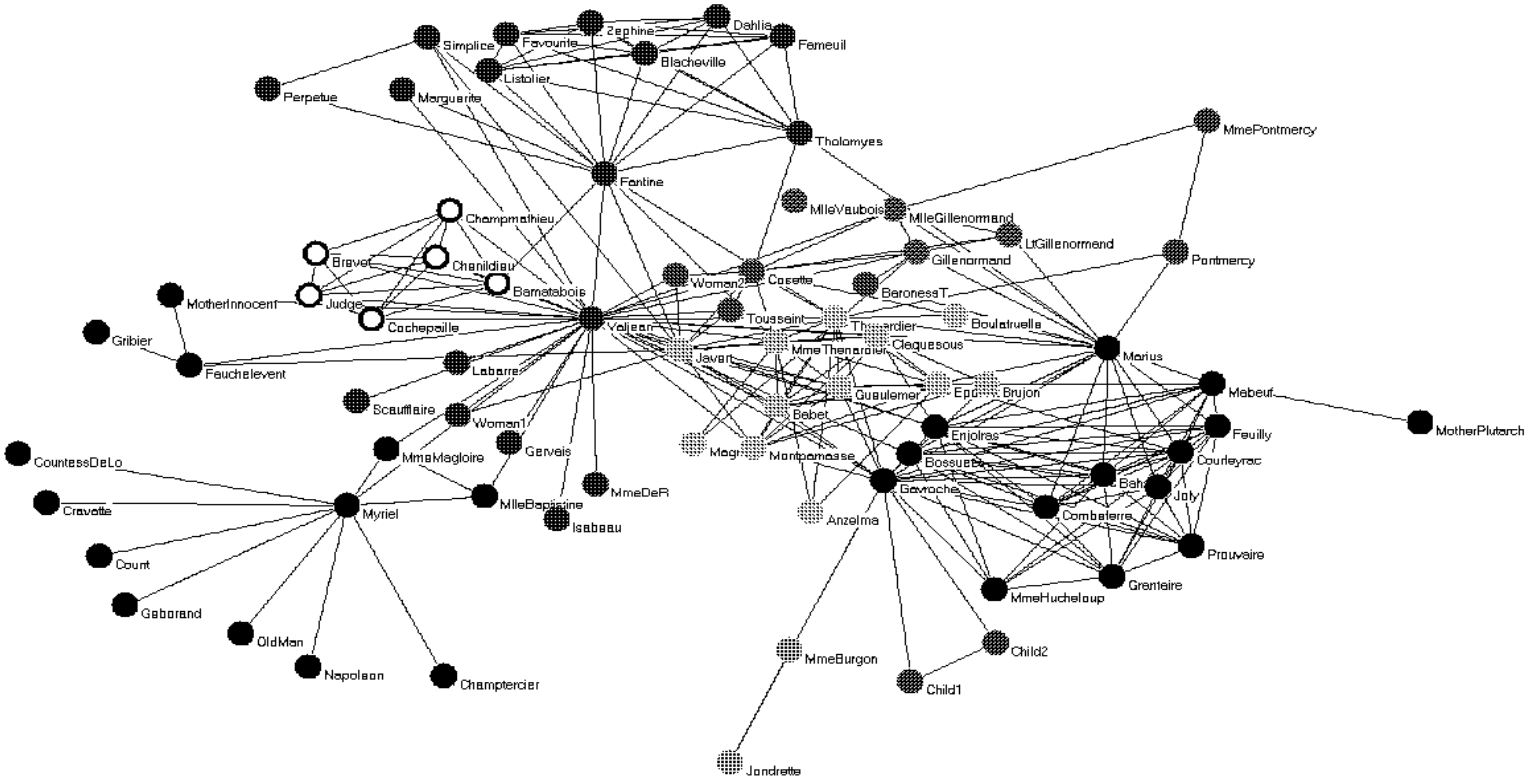

Figure 3: Our Algorithm with Cleaning on the Les Miserables Data

With the time we had left, our group tested the algorithm on a small data set involving the *Prelude* of the $3^{rd}$ Suite of the *Bach Suites for Cello.* Much like text has transitions from paragraph to paragraph, music has phrasings where the character of the music changes. We selected a passage where such a change occurs and decided to see if our algorithm could partition notes used before and after the transition. In Figure 4, vertices are used to represent those notes playable on the cello; hence, $G1$ denotes the lowest g-note available on the cello, $G2$ is the second lowest, and so on. Edges link vertices $u$ and $v$ if the associated notes came immediately before or after one another in the score. Our algorithm found the correct partitioning with parameters $\rho = 200$, $l = 11$, $c = 0.89$, $\lambda = 2$. Table 2 provides the results in text format.

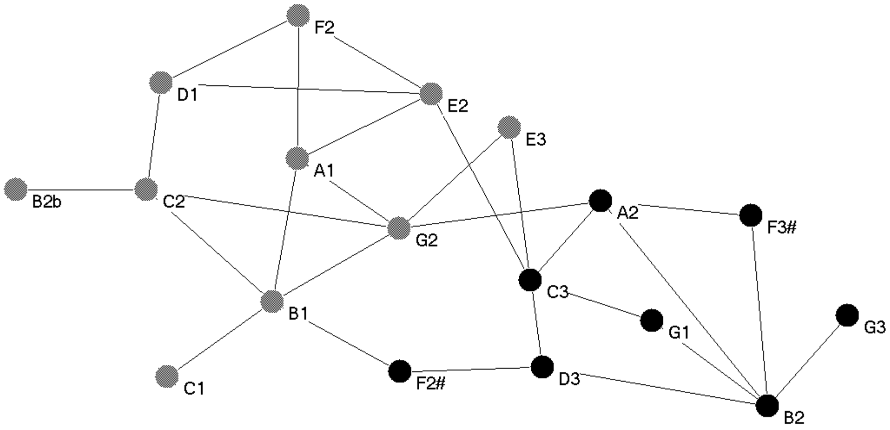

Figure 4: Our Algorithm with Cleaning on the Bach Data

| Bach Communities | |
|---|---|
| Community | Members |
| 1 | A1, B1, B2b, C1, C2, D1, E2, E3, F2, G2 |
| 2 | A2, B2, C3, D3, F2, F3, G1, G3 |

Table 2 - The two communities found by our algorithm when tested on the Bach data.

# 4 Conclusion and Future Work

Our algorithm performed surprisingly well given its simplicity. We believe that there are the following directions for future work:

1. We would like to remove some of the parameters. In particular, having the algorithm be able to set its own $c$ value would be a valuable improvement. We would also like to avoid the use of the clean-up phase. Perhaps there is a way for our algorithm to perform equally well that does not require a specified $\lambda$.

2. A mathematical analysis of the algorithm. It would be nice to have a rigorous treatment of biased intersecting random walks to see what structural property of

the graph this algorithm is actually uncovering. Is the stochastic nature of the algorithm an advantage or a disadvantage? Perhaps a mathematical treatment would allow us to determine whether, given a fixed problem instance, the varying results provided by the algorithm have merit.

3. Testing of more data sets to see the algorithm's behavior along with varying parameters. This would include a comparison against other community-finding algorithms. In particular, we would like to know whether this algorithm performs faster than the current algorithms authored by [2,3]. One of the nice things about this agent-based approach is that no ant needs to have global knowledge of the graph connectivity. Unlike the calculations required for finding geodesic paths, each ant needs to know only local information in order to complete its walk. We suspect that on large graphs, the local-search aspect (short biased random walks) of our algorithm will permit relatively fast analyses of even very large graphs.

4. We would like to experiment with other simple voting procedures to see if we can obtain one that provides consistently better results.

5. Extend the exploration phase of the algorithm. Instead of having one generation of ants go out and explore the graph, have many generations. Those vertices that get hit could have a higher probability of being a starting point for ants of the next generation. This might reinforce the formation of communities while decreasing the inclusion of those nodes that are traversed more than they should be due to the uniform random starting point selection.

## 5 Acknowledgements

This project was the result of work done during four weeks spent at the Santa Fe Institute's 2004 Complex Systems Summer School program. We thank the Santa Fe Institute as well as the organizers of this event for their help and support. We also thank Mark Newman for answering some of our questions and providing testing data.

## References

[1] M. Girvan and M. E. J. Newman. Community Structure in Social and Biological Networks. *Proc. Natl. Acad. Sci.* USA **99**, 7821-7826 (2002).

[2] M. E. J. Newman. Fast Algorithm for Detecting Community Structure in Networks. *Phys. Rev.* E **69**, 066133 (2004). http://www-personal.umich.edu∼mejn/recentpubs.html.

[3] M. E. J. Newman and M. Girvan. Finding and Evaluating Community Structure in Networks. *Phys. Rev.* E **69**, 026113 (2004).

[4] M. E. J. Newman. Detecting Community Structure in Networks. *Eur. Phys. J.* B **38**, 321-330 (2004).

[5] M. E. J. Newman. Coauthorship Networks and Patterns of Scientific Collaboration. *Proc. Natl. Acad. Sci.* USA **101**, 5200-5205 (2004).

[6] M. E. J. Newman. A Measure of Betweenness Centrality Based on Random Walks. submitted to *Social Networks.*

# Appendix

| Les Mis Communities | | |
|---|---|---|
| Community | Our Algorithm Members | Newman Algorithm Members |
| 1 | Champtercier | Champtercier |
| | Count | Count |
| | Countess DeLo | Countess DeLo |
| | Cravatte | Cravatte |
| | Geborand | Geborand |
| | Mlle Baptistine | MlleBaptistine |
| | Mme Magloire | MmeMagloire |
| | Myriel | Myriel |
| | Napoleon | Napoleon |
| | Old Man | OldMan |
| 2 | Fauchlevent | Fauchlevent |
| | Gribier | Gribier |
| | Mother Innocent | Mother Innocent |
| 3 | Jondrette | Jondrette |
| | Mme Burgon | MmeBurgon |
| 4 | Child 1 | Child 1 |
| | Child 2 | Child 2 |
| 5 | Bahorel | Bahorel |
| | Bossuet | Bossuet |
| | Combeferre | Combeferre |
| | Courfeyrac | Courfeyrac |
| | Enjolras | Enjolras |
| | Feuilly | Feuilly |
| | Gavroche | Gavroche |
| | Grantaire | Grantaire |
| | Joly | Joly |
| | Mabeuf | Mabeuf |
| | Marius | Marius |
| | Mme Hucheloup | Mme Hucheloup |
| | Prouvaire | Prouvaire |
| | Mother Plutarch | |

|  |  |  |
|---|---|---|
| 6 | Baroness T | Baroness T |
|  | Cosette | Cosette |
|  | Gillenormand | Gillenormand |
|  | Lt Gillenormand | Lt Gillenormand |
|  | Toussaint | Toussaint |
|  | Mlle Gillenormand | Mlle Gillenormand |
|  | Mlle Vaubois | Mlle Vaubois |
|  | Woman 2 | Woman 2 |
|  |  | Magnon |
|  |  | Mme. Pontmercy |
| 7 | Anselma | Anzelma |
|  | Babet | Babet |
|  | Brujon | Brujon |
|  | Claquesous | Claquesous |
|  | Eponine | Eponine |
|  | Gueulemer | Gueulemer |
|  | Javet | Javert |
|  | Mme Thenardier | Mme Thenardier |
|  | Montparnasse | Montparnasse |
|  | Thenardier | Thenardier |
|  | Boulatruelle |  |
|  | Magruerite |  |
|  |  | Pontemercy |
| 8 | Blacheville | Blacheville |
|  | Dahlia | Dalhia |
|  | Fameuil | Fameuil |
|  | Fantine | Fantine |
|  | Favourite | Favourite |
|  | Listolier | Listolier |
|  | Marguerite | Marguerite |
|  | Perpetue | Perpetue |
|  | Tholomyes | Tholomyes |
|  | Zephine | Zephine |
|  | Simplice |  |
| 9 | Gervais | Gervais |
|  | Isabeau | Isabeau |
|  | Labarre | Labarre |
|  | Mme DeR | Mme DeR |
|  | Scaufflaire | Scaufflaire |
|  | Valjean | Valjean |
|  | Woman 1 | Woman 1 |
|  |  | Bamatabois |
|  |  | Brevet |
|  |  | Chenildeiu |
|  |  | Cockepaille |
|  |  | Scaufflaire |
|  |  | Simplice |

| | | |
|---|---|---|
| 10 | Bamatabois<br>Brevet<br>Champmathieu<br>Chenildieu<br>Cochepaille<br>Jedge | |
| 11 | Mme Pontmercy<br>Pontmercy | |
| 12 | | Boulatruelle |
| 13 | | Mother Plutarch |